2. I'm not sure I understand the question correctly. In order to create a list of search results, we would always have to "check" every element in the list of products, in which case the complexity scales with the number (n) of elements in the list. In other words, the complexity is O(n) (assuming each "check" is done in constant time). So I don't see how this search could be done in constant time.

To do each individual "check" in constant time, assuming we are searching for whole words in each title and not all possible substrings, hashing comes to mind. I'm not sure of an exact implementation, but each unique, space-seperated word could be mapped to a unique index using a hash function. Each index would be pointing to a value, indicating whether or not the word exists in the string. Something similar to the implementation of a Python dictionary.

In my implementation of product search, I use the ".includes()" method, which checks every possible substring. I guess it would be possible to create a similar structure as described above, from each unique substring and use hashing to look them up in constant time, but I don't know how .includes() is actually implemented, and what I just described seems like a lot of work, once the string reaches a certain length.

I'm curious to know the correct answer to this question, so please let me know.

3. That depends on who I'm drinking with ;)